

Flutter ile Sınırları Aşın: Yerelleştirme ve Uluslararasılaşdırma

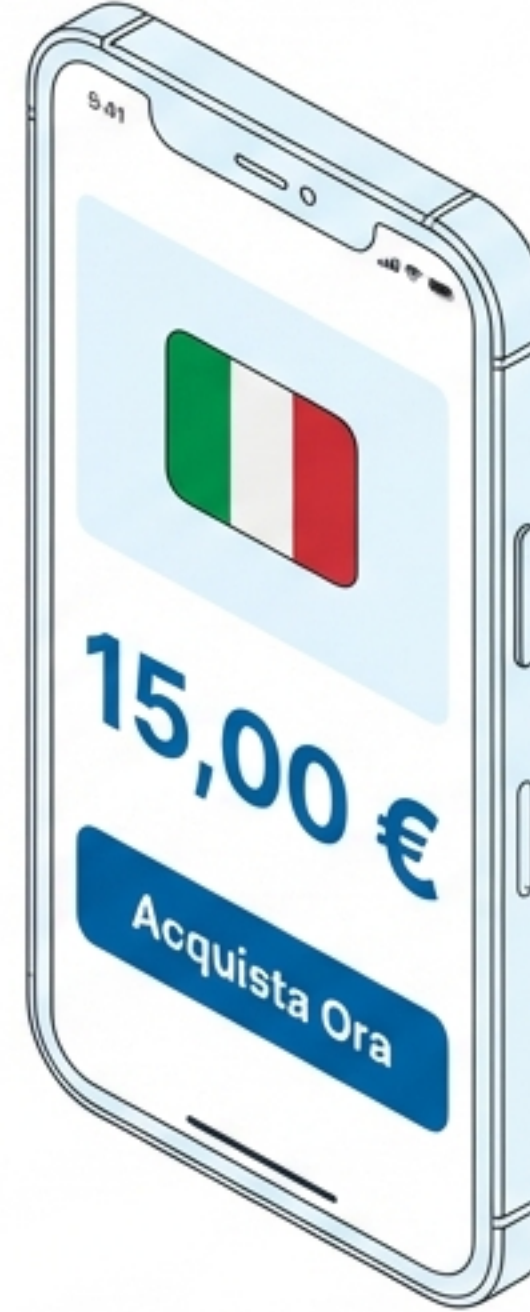
Küresel Kullanıcı Deneyimi için
Teknik Bir Rehber



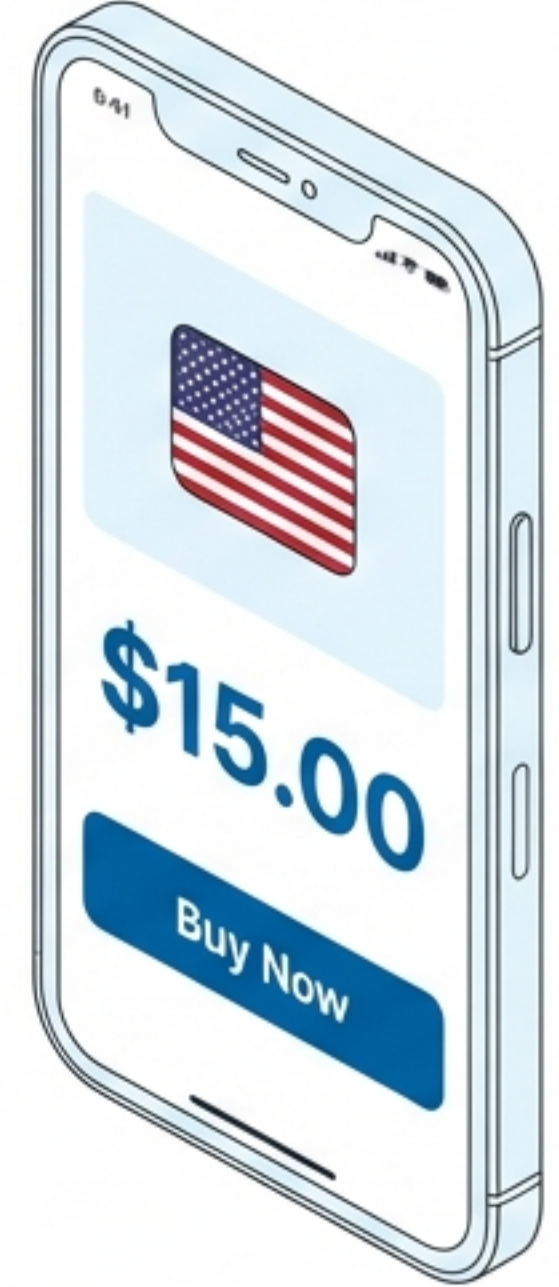
Kullanıcı Deneyimi: İki Farklı Dünya

"Responsive" tasarım sadece ekran boyutuna uyum sağlamak değildir; kullanıcının kültürüne ve coğrafyasına uyum sağlamaktır.

Uygulama, cihazın yerel ayarlarına (locale) göre otomatik olarak 'şekil değiştirir.'



JetBrains Mono
Locale: it_IT



JetBrains Mono
Locale: en_US

Yerelleřtirmenin Temel Tařları

Bir uygulama sınırları ařtıęında deęiřmesi gereken drt temel unsur vardır:



Cmle evirisi

Metinlerin yerel dile uygun řekilde evrilmesi.



Para Birimi

Fiyatların doęru sembolle (\$, €, £) gsterilmesi.



Zaman Formatı

12 saat (8.30pm) veya 24 saat (18.30) tercihleri.



Zaman Dilimleri

Saat bazlı iřlemlerin coęrafi konuma gre ayarlanması.

Araç Seti: package:intl

Tamamen yerelleştirilmiş bir uygulama yapmak zordur; uygulama büyüdükçe karmaşıklık artar. Flutter, bu süreci yönetmek için kapsamlı 'intl' paketini sunar.

Bu paket, cihazın kültürel ayarlarını otomatik olarak algılar ve birçok yerelleştirme aracını beraberinde getirir.

```
import 'package:intl/intl.dart';  
  
void main() => runApp(const DemoApp());
```

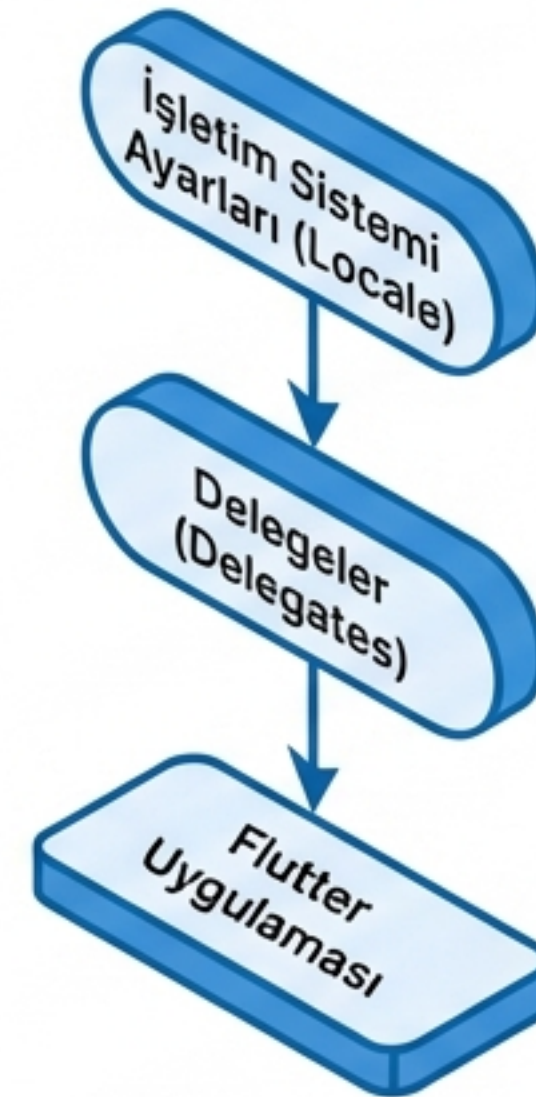


intl

Delegeler ve Yapılandırma

'intl' paketinin çalışması için gereken minimum kurulum, **MaterialApp** içinde '**delegelerin**' (delegates) tanımlanmasıdır. Bu delegeler, işletim sistemi ile uygulama arasında bir **tercüman** gibi çalışır.

```
return MaterialApp(  
  localizationsDelegates: [  
    GlobalMaterialLocalizations.delegate,  
    GlobalCupertinoLocalizations.delegate,  
    GlobalWidgetsLocalizations.delegate,  
  ],  
  // ...  
);
```



Bu kod bloğu, uygulamanın cihazın dil ve bölge ayarlarını otomatik olarak toplamasını sağlar.

Otomatik Özellikler: Para Birimi

Para birimi sembollerini asla sabit kodlamayın (hard-code). `NumberFormat.simpleCurrency` yöntemi, cihazın konumuna göre doğru sembolü getirir.

```
var localeFormat = NumberFormat.simpleCurrency(  
    locale: Localizations.localeOf(context)?.toLanguageTag() ?? 'en',  
);  
debugPrint('${localeFormat.currencySymbol}');
```

Locale: it (İtalya)



Output: EUR

Locale: en_US (ABD)



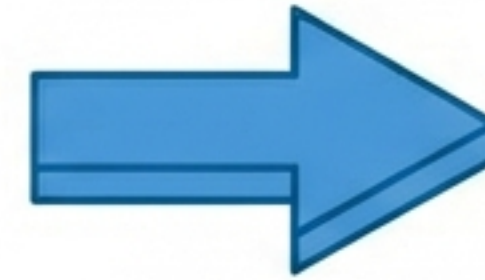
Output: USD

Zamanın Dili: Tarih Formatı

Tarih formatları kültüre göre değişir. DateFormat sınıfı, hem kalıpları (pattern) hem de dil çevirilerini yönetir.

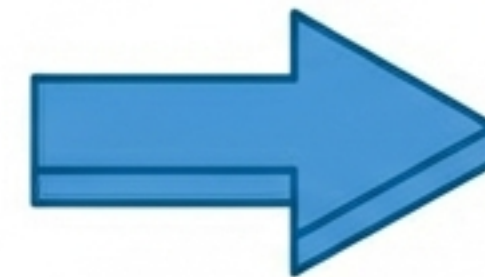
'Sun' (Sunday) otomatik olarak **'dom' (Domenica)** şekline dönüşür. dönüşür. Kullanıcının zihinsel modeline uyum sağlanır.

```
// Varsayılan (en)  
DateFormat('EEE, M/d/yyyy').format(today);
```



Sun, 5/31/2020

```
// İtalyanca (it)  
DateFormat('EEE, M/d/yyyy', 'it').format(today);
```



dom, 5/31/2020

Hazır Arayüz Çevirisi

Flutter'ın yerleşik widget'ları (Material ve Cupertino) çok dillidir. Örneğin, bir 'Arama' butonu için metni elle yazmak yerine, MaterialLocalizations sınıfını kullanabilirsiniz.

Kötü Uygulama (Sabit Kod)

```
child: const Text('Search')
```

Locale: en_US (ABD)



Locale: it (İtalya)



İyi Uygulama (Dinamik)

```
child: Text(MaterialLocalizations.  
  of(context).searchFieldLabel)
```

Locale: en_US
(ABD)



Locale: it
(İtalya)

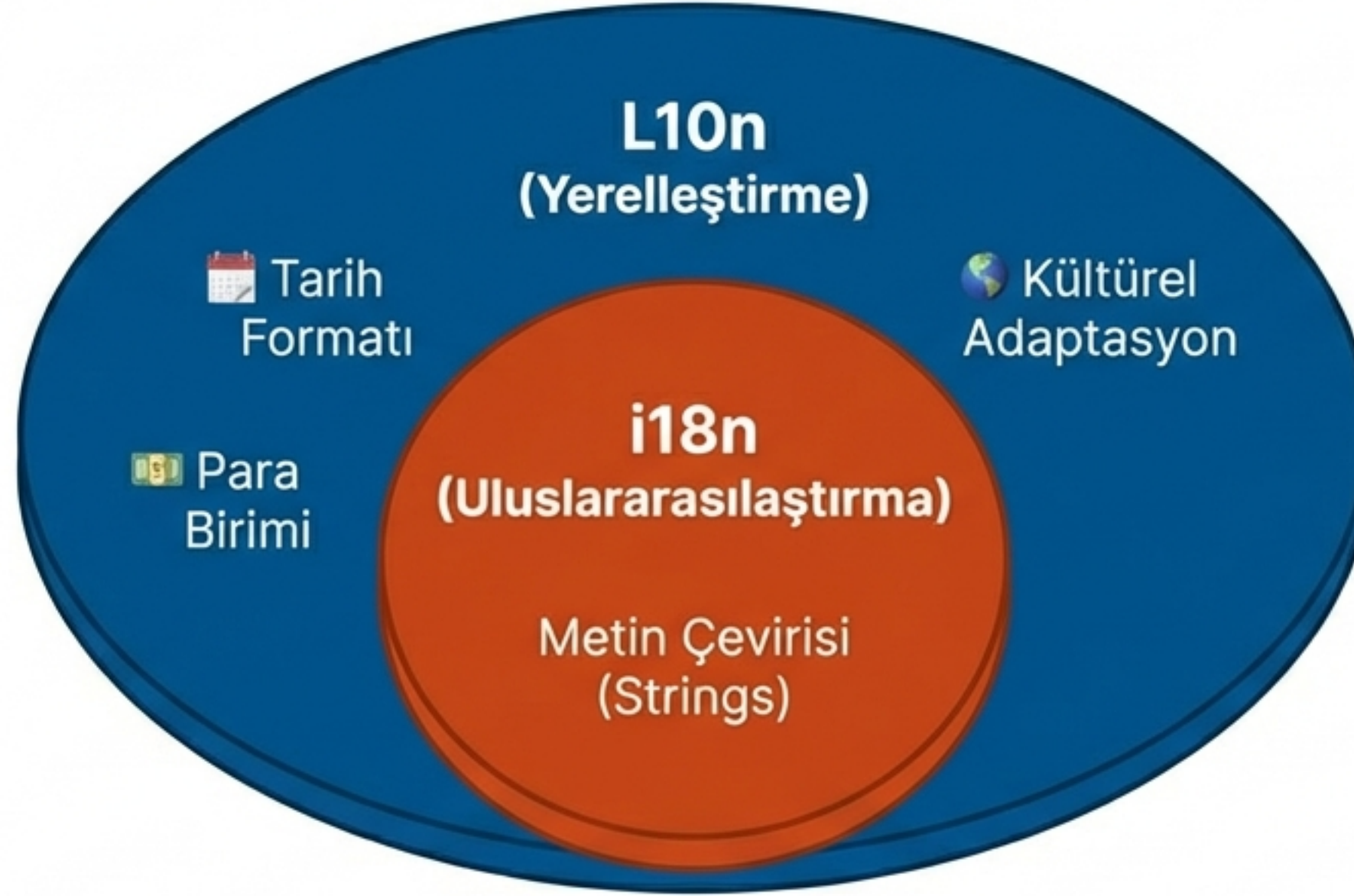


Locale: fr
(Fransa)



Kavram Karmaşası: L10n vs. i18n

Bu terimler eş anlamlı değildir. Biri süreci, diğeri ise şemsiye kavramı ifade eder.



Her i18n işlemi bir L10n tekniğidir, ancak her L10n işlemi sadece metin çevirisi değildir.

Inter Tight

“Özel Metinlerin Zorluğu”

Standart widget’lar ve formatlar tamam. Ancak uygulamanızın kendine özgü başlıkları, açıklamaları ve buton metinleri ne olacak?

Sorun: Flutter’ın standart kütüphanelerine ait olmayan her bir metin (‘String’) için tam çeviri yapabilen bir yapı kurmalısınız.

Hedef: Uygulama açıldığında, kullanıcı dil seçmek zorunda kalmamalı; uygulama cihazın dilini tanıyıp tüm özel içeriği buna göre sunmalıdır.



İki Stratejik Yaklaşım

1. Manuel Uluslararasılaştırma

Temel Yaklaşım

- Daha hızlı uygulanır.
- Kullanımı daha kolaydır.
- Sadece basit metin çevirisi yapar.



2. Intl Paketi ile Uluslararasılaştırma

Gelişmiş Yaklaşım

- Kurulum ve yapılandırma gerektirir.
- Çoğul ekleri (plurals), cinsiyet zamirleri ve gramer kurallarını yönetebilir.
- Büyük ölçekli projeler için idealdir.



Özet ve En İyi Uygulamalar



Sabit Kodlamadan Kaçının

Metinleri asla kodun içine gömmeyin.



Performans

İki teknik de (Manuel ve Intl) performans açısından iyidir; biri diğerinden üstün değildir.



Seçim

İhtiyacınıza en uygun olanı seçin. Emin değilseniz, hangisinin iş akışınıza uyduğunu görmek için her iki yaklaşımı da deneyin.

Kaliteli bir uygulama sadece responsive değil, aynı zamanda yereldir.